Online Database Support Experiences

ods-dba@fnal.gov

Diana Bonham, Dennis Box, Anil Kumar, Julie Trumbo, Nelly Stanfield

Consider Database Needs

Any database server structure can be divided into

- internal (tables, constraints, etc)
- internal memory (shared memory / processes)
- external to the database (physical files)

Learning Curve

Distributed databases vs

1 Database and Many applications

Physical/Logical Structures

1. Internal database structures - a very brief overview of all Oracle internal structures:
    a. Tables, columns, constraints
        a. Breaking up larger tables into smaller ones based on particular value
        b. Helps performance because optimizer will know where to look for particular entries
    b. Datatypes – Specialty
        a. Lobs – Character & Binary
    c. Partitions
        a. Partitioning both Data & indexes

Physical/Logical Structures

Relationships

Indexes, clusters, & hash clusters

Views

Sequences

Stored Procedures, functions, packages, and triggers.

Synonyms

Users, Privileges and roles

Database links

Consider Database Needs

- Scalability
- Performance
- Usage Patterns
- Access
- Parallel query ability
- Uptime Estimates

Consider Database Needs

- Replication (updateable or read only)
- Remote Sites
- Backups
- Recoverability
- Minimize loss of data - rollbacks

Requirements

- Database Liaison position, especially an active liaison with some authority has been helpful.
- Database Schema Evolution – Design Tool
- Application Evolution
- Development, Integration and Production Environments
  - Integration instance - this separation allows changes to be made to any of these areas without effecting the others, and allows testing of all the applications together -- Consider the importance of your testing approach again follow dev -> int -> prd methodology. The only instances of lost and/or corrupted production data has been a result of not testing or insufficient testing in integration
- Test through all phases of development, Integration & Production

Requirements

- Space – Lots of DISK (60% of our problems)
- Up front – design needs to NOT ONLY think about data / indexes
  - Overhead
  - Feature creep
  - DB Evolution
- Plan for current capacity and anticipated growth well ahead of schedule
- Space – Lots of DISK

Requirements

- The performance goals for the system must be identified
- The disk mirroring architecture must be known
  - Automatically doubles disks need improves uptime
- Disks must be dedicated to the database
- Disks must be dedicated to software & log files
- Identical Machines for testing (load rates, inserts, etc)

Requirements

- Root Cause Analysis done whenever there is an unexpected problem. Recommended actions need to be implemented and followed up.
- Simulation testing for sizing, and loading statistics

Daily tasks

- Database(s) are monitored using a commercial monitoring tool as well as our in house tools.
  - Monitor health of database
  - Status of database reads, #of processes, space utilization, etc.
- Create charts for space and cpu usage.
- These tools are also used for decision making regarding tuning / health of the database.

Designer Tool

- Entity Relationship diagrams have proved essential in documenting and showing users the logic behind an application.
- They have also proved to be essential in defining space needs.
- Following Standards - cutting scripts
- Ddl and all code stored in source code

  Document all schemas
- Application owners worked both with physicists and dba's using a DESIGN TOOL

Recovery/Backup

- Very large databases have special backup and recovery needs. Database uptime percentage and mean time to recovery must be reasonable based on database size and dollars available to implement recovery and failover methods.
- Costs escalate quickly when short mean times and high uptimes are required.

Recovery/Backup

- UPS
- Disks
- Tape
- Robot/Library – depending on size of database

- Hardware failover
  - Disks
  - Cpu
- Warehouse
- Standby Database

Application and Database Maintenance

- Upgrade plan to ensure database & application evolution
- We have had to upgrade database software
    1) obtain new and needed capabilities
    2) security reasons
- Application new features being addressed
- Bug fixes

Lessons Learned

- **Space**
- Disk Estimates –
  - Overhead, Infra
- Database Dictator
- Local People at
  - Experiment

- **Design Tools**
- Best designs come from a close working relationship with application, physicist and the dba

Lessons Learned

- Establish Standards
  - http://fncduh1.fnal.gov/supportdb/working/run2_standards.html
- **Testing in integration before going to Production**
- Using Designer Tool Constraints relationships,indexes
  - (data checking)
- Database experts – 1 is not enough, plan for personnel changes, cross-training much easier if standards followed

Lessons Learned

- Database upgrade might affect application
- Amount of training based on architecture implemented
- Learning Curve
- Backups VLD
- Root Cause – *need time to implement what was learned*
- Documentation
- **Space**